Interfacing the Am79C30A to Non-iAPX Microprocessors

Advanced Micro Devices
Digital Terminal Product Applications

The Am79C30A DSC microprocessor interface was designed for connection to a non-multiplexed iAPX bus with minimal external logic. This can pose design challenges when attempting to use a DSC with other microprocessor architectures, such as the Motorola 68K. This application note addresses these challenges and shows specifically how the DSC can be connected to a Motorola 68302 with minimal external logic, though the concepts presented can be applied to any microprocessor interfaced with the DSC.

Interfacing the Am79C30A DSC to a microprocessor is an easy task, or is it? Depending on the microprocessor, it may appear to be extremely easy or relatively complex. With a few considerations, however, interfaces to most microprocessors are simple.

Design issues that frequently arise when connecting a DSC to a microprocessor are the amount of external logic required and the speed at which the interface is expected to run. These issues tend to correlate to typical system concerns of power, board space, and computing power. Due to the wide variety of applications for which the DSC is suited, the list of microprocessors and controllers which might be used is extensive. They vary from basic controllers such as an 8051 in an ISDN phone to RISC processors in a workstation. This application note will try to point out the key considerations of interfacing to the DSC's MPI block.

The DSC Microprocessor Interface (MPI)

The microprocessor port of the Am79C30A DSC provides the user with an industry standard non-multiplexed bus interface. Separate Address and Data buses; as well as Read (RD*), Write (WR*), and Chip Select (CS*) control lines are provided. This allows for ease of connection to an iAPX class microprocessor with minimal external logic. In a simple system, the DSC can be treated as an I/O device with CS* tied LOW if no other I/O devices exist. Data, address, RD*, and WR* signals are all connected directly with no external logic necessary. Only in more complex systems, with multiple I/O devices, is external logic required. That logic would be needed to generate an appropriate CS* signal.

A multiplexed environment requires the addition of some more external circuitry such as latches to demultiplex the Address/Data bus, but this should not be a major design issue. Examples of such an application can be found in AMD's Development/Evaluation boards for the DSC. Both the Power Compliant Telephone (PCT) for the 8051 environment and the Terminal Equipment - S-Interface (TE-S) for the 80186 environment are proven design examples.

What about non-iAPX processors? Not all microprocessors or controllers supply the required control signals directly without external logic. Also many systems require processor speeds exceeding those demonstrated on the AMD produced Development boards. Problems arise as the speed of the microprocessor interface increases and the amount of external logic required to generate the necessary control signals increases.

The Microprocessor Read/Write timing section of the DSC data sheet can be used to the designer's advantage in this situation if interpreted properly. The timing parameter definitions are relatively self explanatory and consistent with industry standards, but the "Notes" provide substantial insight into key design considerations. The designer can use these to his advantage.

All of these notes provide insight into the application of the timing parameters, but note 9 provides the most useful information. Note 9 states that ***"RD* and WR* pulse width, Address Setup and Hold, and Data Setup and Hold timing is measured from the points where both CS* and RD* or WR* are low (active) simultaneously"***. This means that timing parameters referring to RD* or WR* pin relationships to other signals are also applicable to CS*. CS* can effectively be substituted for RD* or WR* in these parameters with no change to timing. The only exception is "Address Hold after WR* High", which is 12ns instead of 10ns. A timing parameter such as "Address Hold after RD* High" might be difficult to meet, while "Address Hold after CS* High" may be easier.

Interface Design

As a test case let us go through an interface design and timing analysis based on a 68302. This will address the two concerns of minimizing external logic and operating with higher speed processors.

Figure 1 shows a minimal interface between a 16MHz Motorola 68302 and an AMD Am79C30A DSC. The 68302 provides a non-multiplexed data bus, so address and data lines can be connected directly. You will notice that address lines A3-A1 out of the 68302 are utilized instead of A2-A0. Due to the difference in data bus width between the DSC and the 68302, this allows the 68302 to access the DSC in one byte units.

The 68302 uses a single control line to differentiate between Read and Write operations. The DSC however requires separate control line. Generation of independent RD* and WR* signals can be accomplished with a pair of OR gates and an INVERTER. This simple circuit minimizes gate count and fulfills timing requirements, as will be seen in the timing analysis. LDS* is used in place of CSx* in the generation of RD* to prevent the occurrence of a RD* glitch during Write cycles. Note that this is possible due to the delay in LDS* during Write cycles which does not occur during Read cycles. In the case of WR*, CS* is used instead of LDS* to maximize the WR* pulse width without adding wait states.

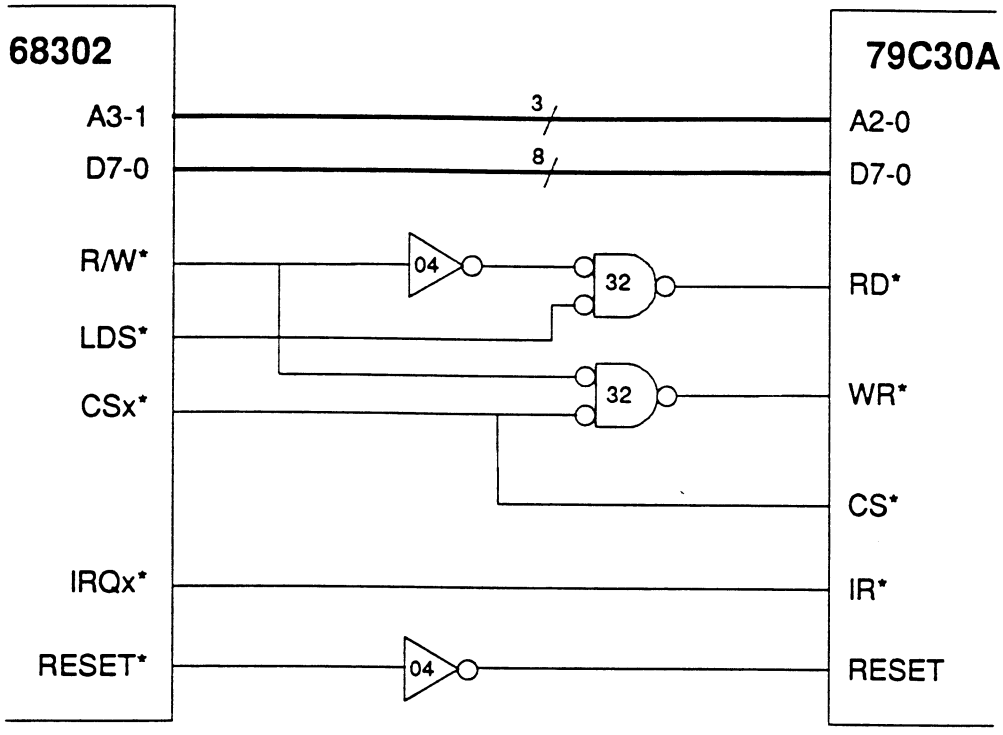

Figure 1

Finally CSx* and IRQx* are connected directly under the assumption that the system is sufficiently simple. RESET* is connected through an INVERTER to compensate for the difference in logic level.

Interface Timing Analysis

Address Setup and Hold time violation is the most frequent microprocessor interfacing problem experienced by DSC users. This is because these parameters are typically the most difficult to meet. By design, high speed processors attempt to minimize such requirements, plus the compounding effects of gate delays due to external interface logic. Both can easily cause violation of these parameters.

The DSC does not internally latch the address pins. It requires that stability on the address lines be assured externally. Symptoms of failure to meet Address Setup and Hold time requirements of the DSC can vary from inability to reliably Read or Write intended registers, to missing status

conditions. Anytime a register access problem is encountered, it is usually a good idea to check these parameters. The DSC user should keep in mind Note 9 discussed previously. By making use of this fact, the designer can avoid Address Setup and Hold violations.

Take as an example the 68302. Since the 68302 uses a single control line to differentiate between Reads and Writes, the user must externally generate the RD* and WR* control signals required by the DSC. This adds gate delays between the 68302 and the DSC. Such delays can cause timing problems, but they can also be used to compensate for other potential violations. Address Setup and Hold time of the 68302 with respect to Chip Select 0-3 (CSx) is 15ns. This is sufficient to meet DSC Address Hold requirements, but not Setup. As mentioned previously, since the Address Setup requirement of the DSC is with respect to RD* or WR* and CS* active simultaneously, the gate delay in the generation of RD* or WR* can be used to assure sufficient Setup time.

Proper operation of the DSC with a 16MHz 68302 will require two Wait states. This is necessary to meet RD* or WR* pulse width requirements of the DSC. Similar requirements exist on the recovery time for these operations, but this should be guaranteed by the instruction fetch Read cycles of the DSC access even if back-to-back operations are attempted.

Verification of the remaining parameters is relatively straight forward. Details of the analysis on a parameter by parameter basis can be found in Appendix A. This analysis relates to the timing diagram provided in Figure 2. Certain assumptions are made (see Appendix A Notes), but all are considered reasonable and no special justification should be required.

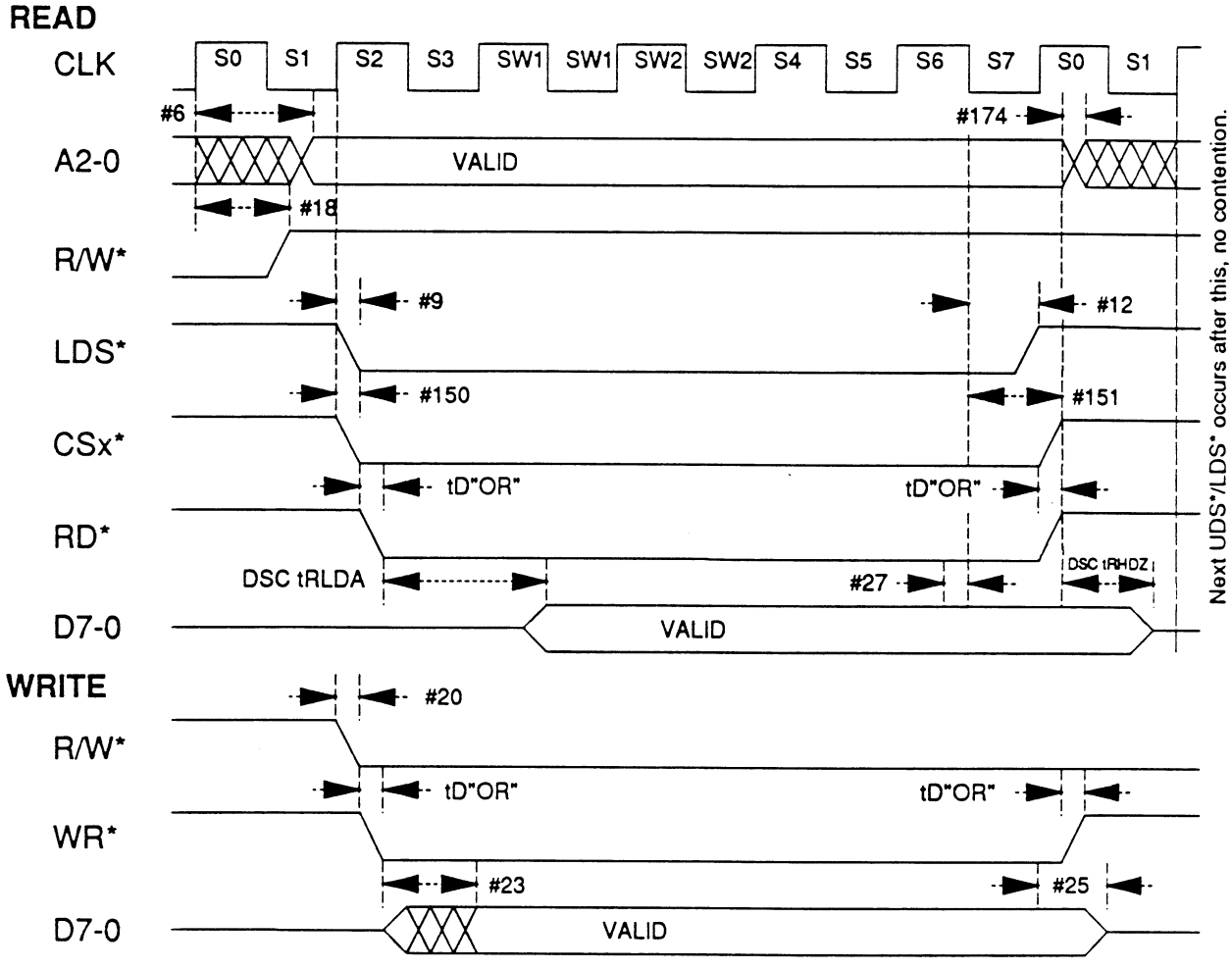

Figure 2

Summary

There is nothing difficult about interfacing the DSC to any microprocessor. It has a relatively simple interface. Access times are slow by today's high-end microprocessors, but with proper interpretation and effective application of the timing parameters, one can both minimize interface logic and run at interface speeds beyond the original design goals of the DSC.

Appendix A:

Note: A)#X refers to timing numbers specified in the 68302 data sheet.
B)"wc" = worst case & "bc" = best case.
C)CS indicates CS bar.*
D)Loading of less than 50pf is assumed on all outputs.
E)Analysis assumes 74HC/HCT class logic: max delay = 25ns, min = 5ns.
F)Analysis based on a 16MHz 68302 operating in IMP bus master mode.

READ:

1. DSC t_{RLRH}: RD* pulse width, Min = 200ns.

   Req: S2-(Assert delay)$_{wc}$+S3+SW1+SW2+S4+S5+S6+(Negate delay)$_{bc} \geq$ 200ns

   Act: CS* asserted delay = #150$_{max}$ = 40-4 = 36ns
   #150 = Clock High to CS*, IACK* Low
   RD* asserted delay = #9$_{max}$+$t_{D\text{-}OR\text{-}max}$ = (30-4)+25 = 51ns
   #9 = Clock High to AS*, DS* Asserted
   Worst case is RD* asserted delay, assume 51ns

   CS* negated delay = #151$_{min}$ = 0ns
   #151 = Clock Low to CS*, IACK* High
   RD* negated delay = #12$_{min}$+$t_{D\text{-}OR\text{-}min}$ = (0+5) = 5ns
   #12 = Clock Low to AS*, DS* Negated
   Best case is CS* negated delay, assume 0ns.

   30-(51)+30+60+60+30+30+30+(0) = 219ns (assumes 2 WAIT states)

2. DSC t_{RHRL}: Read recovery time, Min = 200ns.

   Read and write cycles to the DSC will be separated by memory accesses, therefore this requirement will be met by default.

3. DSC t_{AVRL}: Address valid to RD* Low, Min = 20ns.

   Req: (Address Setup before Control Asserted)$_{bc} \geq$ 20ns

   Act: #173$_{min}$ = 15ns is insufficient, but:
   #173 = Address, FC Valid to CS* Asserted

   Address asserted delay related to S2 = S0+S1-#6$_{max}$ = 30+30-45=15ns
   #6 = Clock High to FC, Address Valid
   RD* asserted delay = #9$_{min}$+$t_{D\text{-}OR\text{-}min}$ = 3+5 = 8ns
   #9 = Clock High to AS*, DS* Asserted

15+8 = 23ns

4. DSC t_{AHRH}: Address hold after RD* High, Min = 10ns.

Req: (Address Hold after Control Negated)$_{wc} \geq$ 10ns

Act: #174$_{min}$ = 15ns
#174 = CS* Negated to Address, FC Invalid

5. DSC t_{RHCH}: RD* High to CS* High, Min = 0ns.

DSC allows CS* to go High before RD*.

6. DSC t_{RACC}: Read access time, Max = 80ns.

Req: S2-(Asserted Delay)$_{wc}$+S3+SW1+SW2+S4+S5+S6-(Data Delay)$_{wc} \geq$ #27
#27 = Data-In Valid to Clock Low (Setup Time on Read)

Act: CS* asserted delay = #150$_{max}$ = 40-4 = 36ns
#150 = Clock High to CS*, IACK* Low
RD* asserted delay = #9$_{max}$+$t_{D*OR*max}$ = (30-4)+25 = 51ns
#9 = Clock High to AS*, DS* Asserted
Worst case is RD* asserted delay, assume 51ns.

Data delay = t_{RACC} = 80ns

30-(51)+30+60+60+30+30+30-(80) = 139ns $\geq$ 7ns

7. DSC t_{RHDZ}: RD* or CS* High to data Hi-Z, Max = 50ns.

Req: Data bus must be Hi-Z before DS* of next cycle asserts possibly allowing contention. Data hold time requirement of 68302 (#29 & #178) is 0ns.
(Data Hi-Z) < (Data Possible of Next Cycle)

Act: CS* negated delay = #151$_{max}$ = 40-4 = 36ns
#151 = Clock Low to CS*, IACK* High
RD* negated delay = #12$_{max}$+$t_{D*OR*max}$ = (30-4)+25 = 51ns
#12 = Clock Low to AS*, DS* Negated
CS* negates first, so assume 36ns.

Data Hi-Z therefore equals #151$_{max}$+t_{RHDZ}.

Data in next cycle not possible until DS* asserted, S7+S0+S1+#9$_{min}$.
#9 = Clock High to AS*, DS* Asserted

36+50 = 86ns < 30+30+30+3 = 93ns

1. DSC t_{RDCS}: RD* Low to CS* Low, Max = 30ns.

   CS* goes Low before RD*, parameter only applies to CS* going Low after RD*.

WRITE:

1. DSC t_{WLWH}: WR* pulse width, Min = 200ns.

   Req: S2-(Assert delay)$_{wc}$+S3+SW1+SW2+S4+S5+S6+(Negate delay)$_{bc} \geq$ 200ns

   Act: CS* asserted delay = #150$_{max}$ = 40-4 = 36ns
   WR* asserted delay = #150$_{max}$+$t_{D\text{-}OR\text{-}max}$ = 36+25 = 51ns
   #150 = Clock High to CS*, IACK* Low
   Worst case is WR* asserted delay, assume 51ns.

   CS* negated delay = #151$_{min}$ = 0ns
   WR* negated delay = #151$_{min}$+$t_{D\text{-}OR\text{-}min}$ = (0+5) = 5ns
   #151 = Clock Low to CS*, IACK* High
   Best case is CS* negated delay, 0ns.

   30-(51)+30+60+60+30+30+30+(0) = 219ns (assumes 2 WAIT states)

2. DSC t_{WHWL}: Write recovery time, Min = 200ns.

   Read and write cycles to the DSC will be separated by memory accesses, therefore this requirement will be met by default.

3. DSC t_{AVWL}: Address valid to WR* Low, Min = 20ns.

   Req: (Address Setup before Control Asserted)$_{bc} \geq$ 20ns

   Act: #173$_{min}$ = 15ns is insufficient, but:
   #173 = Address, FC Valid to CS* Asserted

   Address asserted delay related to S2 = S0+S1-#6$_{max}$ = 30+30-45=15ns
   #6 = Clock High to FC, Address Valid
   WR* asserted delay = #150$_{min}$+$t_{D\text{-}OR\text{-}min}$ = 0+5 = 5ns
   #150 = Clock High to CS*, IACK* Low

   15+5 = 20ns

4. DSC t_{AHWH}: Address hold after WR* High, Min=10ns.

   Req: (Address Hold after Control Negated)$_{wc} \geq$ 12ns (if based on CS* negated)

   Act: #174$_{min}$ = 15ns
   #174 = CS* Negated to Address, FC Invalid

5. DSC t_{WHCH}: WR* High to CS* High, Min = 0ns.

   DSC allows CS* to go High before WR*.

6. DSC t_{DSWH}: Data setup to WR* High, Min = 100ns.

   Req: S3-(Asserted Delay)$_{wc}$ +SW1+SW2+S4+S5+S6+(Negated Delay)$_{bc} \geq$ 100ns

   Act: Data asserted delay = #23$_{max}$ = 30ns
   #23 = Clock Low to Data-Out Valid
   CS* negated delay = #151$_{min}$ = 0ns
   #151 = Clock Low to CS*, IACK* High

   30-30+60+60+30+30+30-0 = 150ns

7. DSC t_{DHWH}: Data hold after WR* High, Min = 10ns.

   Req: (Data Hold after Control Negated)$_{wc} \geq$ 10ns

   Act: #172$_{wc}$ = 10ns
   #172 = CS* Negated to Data Out Invalid (Write)

8. DSC t_{WRCS}: WR* Low to CS* Low, Max = 30ns.

   CS* goes Low before WR*, parameter only applies to CS* going Low after WR*.